Podaci i njihova namena

U ovom kursu biće objašnjeno gde i na koji način se skladište podaci na računaru i kako se njima može manipulisati. Kurs će, pored upoznavanja sa analizom podataka i regresijom, obuhvatiti i rad sa relacionim i nerelacionim bazama podataka, rad sa redovima poruka, kao i rad sa podacima pomoću ORM rešenja, od kojih je najpopularnije SQLAlchemy.

Sve što u određenom zapisu predstavlja ideju, činjenicu ili događaj nazivamo podatkom. Podaci se mogu koristiti za opisivanje nekog entiteta, izolovane činjenice ili opažanja nastalog posmatranjem nekog procesa. Osnovna namena podataka ogleda se u njihovom korišćenju za interpretaciju, obradu i komunikaciju, kako od strane ljudi tako i od strane mašina.

Kao što smo mogli da primetimo u jednom od prethodnih kurseva, programski jezik Python ima više ugrađenih tipova podataka, koji se međusobno razlikuju po zapisu i nameni:

- numerički tip podatka: *int*, *float*, *complex*;
- tekstualni tip podatka: *str*;
- logički tip podatka: *bool*;
- binarni tip podatka: *bytes, bytearray, memoryview*;
- sekvenca: *list, tuple, range*;
- rečnik: *dict*;
- set: *set, frozenset*.

Od tipa podatka zavisi i način njegovog skladištenja, kao i skup operacija koje je nad njim moguće obavljati. Svaka aplikacija zahteva različite strategije skladištenja podataka; stoga je veoma važno u obzir uzeti faktore kao što su količina podataka, učestalost njihovog korišćenja i odabir tipa podatka koji ih najpreciznije definiše.

Baze podataka

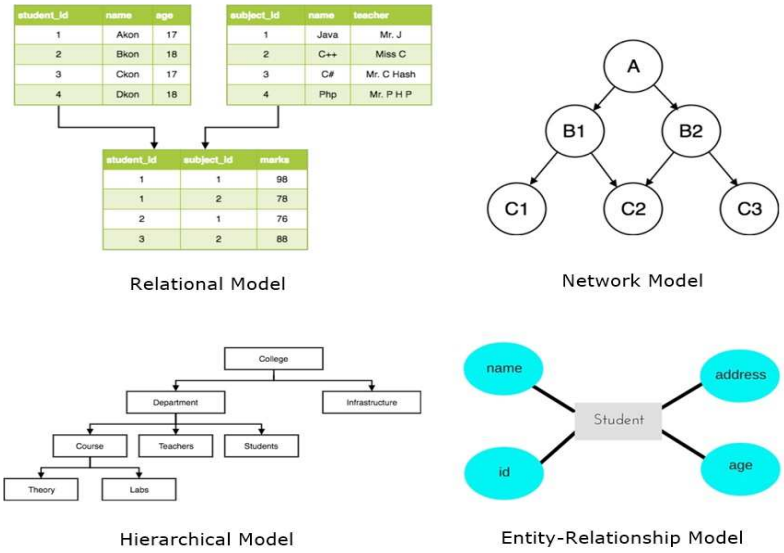
Radi bolje organizacije i veće pristupačnosti, poželjno je da se srodni podaci skladište na jednom mestu. U tu svrhu koriste se baze podataka. **Baza podataka** predstavlja zbirku sistemski skladištenih podataka uređenu u cilju jednostavnijih odgovora na upite sistema. Podaci koji se skladište u bazi predstavljeni su u vidu tabela koje razlikuju redove i kolone. Ovi podaci skladišteni su u spoljnoj memoriji računara i dostupni su različitim korisnicima i aplikacionim programima.

Prednost ovakvog načina skladištenja je korišćenje zajedničkog softvera upotrebom logičke strukture baze, što znači da fizičko poznavanje podataka nije neophodno da bi se njima moglo upravljati. Logička struktura podataka zasniva se na njihovoj logičkoj povezanosti, a ne na njihovom fizičkom skladištenju na medije.

Modeli podataka

Za prikaz logičke strukture podataka koristi se neki od modela podataka. **Modeli podataka** služe za koncipiranje, projektovanje i implementaciju baze podataka. Ti modeli mogu biti relacioni, mrežni, hijerarhijski ili objektni.

- **Relacioni model** je najzastupljeniji model koji se koristi za modeliranje realnih problema i dinamičko upravljanje podacima. Ovaj model objašnjava odnose između podataka u bazi koristeći matematički pojam relacije.
- **Mrežni model** predstavljen je u vidu usmerenog grafa čiji čvorovi predstavljaju tip zapisa, a putanje tih čvorova veze između njih.
- **Hijerarhijski model** zasniva se na odnosu *nadređeni–podređeni*. Prikaz ovog modela bazira se na jednom stablu ili na skupu stabala. Ovaj model je nastao iz mrežnog i predstavlja njegov podtip.
- **Objektni model** izgrađen je na konceptima objektno orijentisanog programiranja. Bazu podataka čini skup trajnih objekata koji imaju svoje atribute i metode. Svaki od objekata pripada nekoj od klasa koje između sebe mogu uspostaviti vezu nasleđivanja, agregacije i korišćenja.



Slika 1.1 Modeli podataka[1]

Sistem za upravljanje bazama podataka (SUBP)

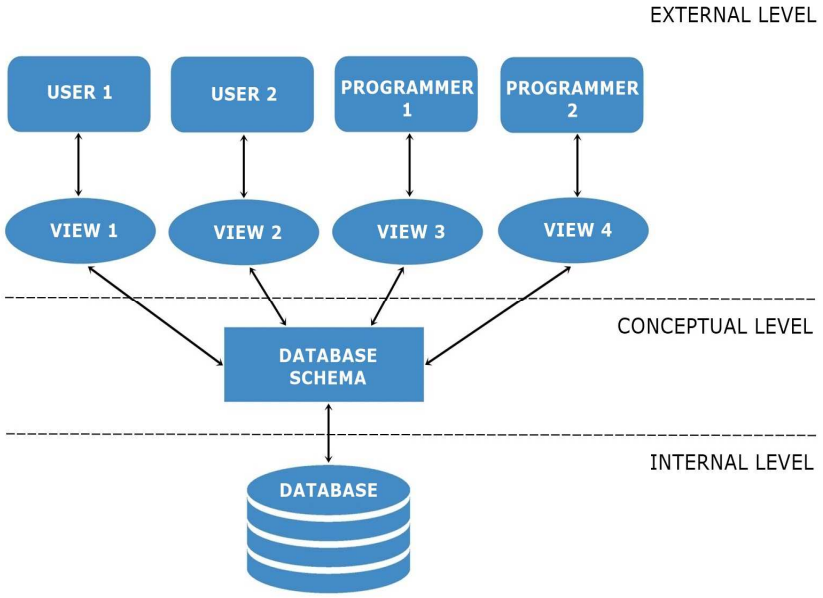
Za efikasno čuvanje i pretraživanje korisničkih podataka iz baze uz poštovanje određenih bezbednosnih mera zadužen je **sistem za upravljanje bazama podataka – SUBP** (database management system – DBMS). Njegova funkcija ogleda se u fizičkom prikazu podataka u skladu sa odabranom logičkom strukturom, odnosno modelom. Ovaj softverski sistem eliminiše nedostatke stare, klasične obrade podataka, koja se zasnivala na sistemu upravljanja datotekama.

[1] https://www.studytonight.com/dbms/database-model.php

Prednosti novog SUBP ogledaju se u sledećim karakteristikama:

- **realni entiteti** – koristi subjekte iz stvarnog sveta za dizajniranje svoje arhitekture, uključujući i njihova stanja i ponašanja;
- **tabele zasnovane na relacijama** – omogućava formiranje tabela odnosa između entiteta u sistemu;
- **izolacija podataka i aplikacija** – baze podataka predstavljaju aktivni entitet u odnosu na same podatke, koji su pasivni;
- **manje redundantnosti** – ovaj SUBP sledi pravila normalizacije, koja odvaja relaciju svaki put kada se neki od atributa ponavlja;
- **doslednost** – odnosi se na konzistentnost relacija između entiteta;
- **upitni jezik** – ovaj jezik olakšava preuzimanje podataka i manipulaciju podacima;
- **ACID osobine** – sledi koncept atomičnosti, konzistentnosti, izolacije i trajnosti (atomicity, consistency, isolation and durability); ovi koncepti se primenjuju na transakcije koje manipulišu podacima u bazi podataka i služe za to da održe bazu podataka u slučaju istovremenog delovanja više transakcija ili u slučaju otkaza; ove osobine će biti detaljno obrađene u daljem toku kursa;
- **višekorisnički i istovremeni pristup** – omogućen je pristup više korisnika, kao i istovremena obrada podataka;
- **višestruki pogledi** – više načina pregleda za različite korisnike – korisnik može izabrati da vidi samo informacije koje su mu potrebne;
- **sigurnost** – odnosi se na ograničenja u pogledu unošenja podataka i pristupa podacima.

Radi boljeg razumevanja rada sa bazama podataka napravljena je generalna podela SUBP po nivoima na osnovu kriterijuma korisničkog pristupa. **Arhitektura sistema za upravljanje bazama podataka (ANSI/SPARC arhitektura)** razlikuje tri nivoa: interni, konceptualni i eksterni (slika 1.1).



Slika 1.2. Arhitektura Sistema za upravljanje bazama podataka (ANSI/SPARC)

- **Interni (fizički) nivo** prikazuje kako su podaci raspoređeni u spoljnoj memoriji. Podaci se na ovom nivou mogu razložiti na više nivoa apstrakcije, koji mogu biti konkretne putanje ili datoteke. Ovaj nivo vide samo programeri koji su razvili SUBP.
- **Konceptualni nivo** definisan je konceptualnom šemom podataka. Na ovom nivou je uređen logički kontekst celokupne baze podataka. Ovako izgleda baza podataka iz ugla projektanta, odnosno sistemskog administratora.
- **Eksterni nivo** – na ovom nivou se definišu svi lokalni tipovi podataka i veze među njima u skladu sa odabranim modelom. Pogled predstavlja zapis jedne lokalne definicije. Ovo je nivo na kojem programeri i drugi korisnici koriste aplikaciju.

Napomena

Pogled ili view u bazama podataka predstavlja jedan od načina prikaza podataka korišćenjem grupe naredbi upućenih serveru. Na osnovu zadate naredbe server prikazuje tačno one podatke koje je korisnik tražio. Pogledi se još zovu i virtuelne tabele. Definisanjem pogleda kreira se kopija originalne tabele čije je podatke, kao i kod originalnih tabela, moguće ažurirati i brisati.

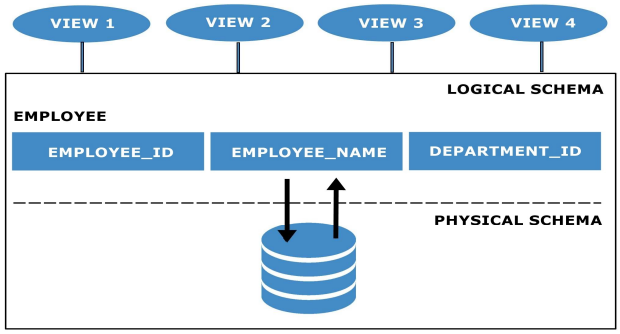
Šema baze podataka

Šema baze podataka je skeletna struktura koja predstavlja logički prikaz celokupne baze podataka. Pomoću šeme dizajneri prikazuju organizaciju podataka u bazi i njihove međusobne odnose (relacije). U okviru šeme baze podataka definisana su i ograničenja koja na podatke treba primeniti. Dizajneri baza podataka kreiraju šeme kako bi pomogli programerima da razumeju bazu podataka i učine je korisnom.

Šeme baze podataka se mogu podeliti u dve kategorije (slika 1.2):

- **fizička šema** – fizičko skladištenje podataka i oblik tog skladištenja (indeksi, fajlovi);
- **logička šema** – definiše tabele, poglede i ograničenja integriteta – ili, drugim rečima, sva logička ograničenja koja treba primeniti na skladištene podatke.

Ova dva sloja svoje funkcije obavljaju izolovano. Ako bismo napravili neke promene u podacima tabele, to ne bi uticalo na podatke skladištene na disku. Isto tako, kada bismo hard-disk zamenili SSD memorijom, logička šema podataka bi ostala nepromenjena.



Slika 1.3. Struktura šeme baze podataka

Pitanje

Označiti tačnu tvrdnju.

- Najzastupljeniji model podataka je mrežni model.
- Najzastupljeniji model podataka je objektni model.
- **Najzastupljeniji model podataka je relacioni model.**

Objašnjenje:

Relacioni model je najčešće korišćeni model podataka još od 1980-ih godina pa sve do danas.

Jezici baza podataka

Za komunikaciju između korisnika i SUBP u aplikaciji koriste se posebni jezici. Te jezike prema nameni možemo razvrstati u tri kategorije:

- **jezik za opis podataka (DDL – data definition language)** – služi za definisanje logičke strukture podataka i veze među njima; ovakve jezike koriste administratori u svrhe zapisa šeme ili pogleda; primeri ovakvih jezika su Pascal, C i PL/I;
- **jezik za upravljanje podacima (DML – data manipulation language)** – koristi se za dodavanje, brisanje i ažuriranje podataka u bazi. Često predstavlja podjezik nekog drugog jezika baza podataka, kao što je SQL;
- **jezik upita (QL – query language)** – ovakav jezik služi za interaktivno pretraživanje baze od strane korisnika; ovi upiti nisu proceduralni, odnosno, ne prikazuju postupak za dobijanje rezultata, već same rezultate.

Sa razvojem tehnologije rasla je i potreba korisnika za integrisanim jezikom baza podataka koji će imati sve tri gore navedene funkcionalnosti. Najzastupljeniji integrisani jezik baza podataka je **SQL (Structured Query Language)**. Ovaj jezik se može koristiti interaktivno, upotrebom nekog interpretiranog okruženja, ili se može uklopiti u aplikacioni program.

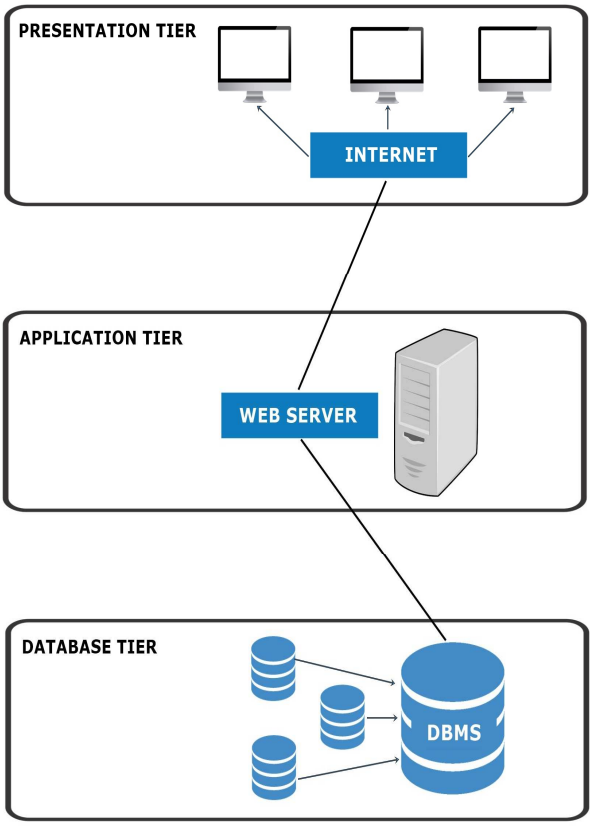
Napomena

SQL je nekada važio za neproceduralni jezik, sve do verzije koja je puštena 1999. godine. Uvođenjem ove verzije podržani su korisnički definisani tipovi podataka, koji predstavljaju jednu od karakteristika objektno orijentisanog programiranja.

Baze podataka i aplikacioni programi

Kako bismo bolje razumeli ulogu baza podataka u okviru jednog aplikacionog programa, neophodno je da definišemo **arhitekturu sa aspekta aplikacije**. U zavisnosti od krajnjeg korisnika i namene, razlikujemo više arhitektura. Jednoslojnu arhitekturu čini samo jedan entitet i to je SUBP. Ova arhitektura je namenjena programerima i dizajnerima kao krajnjim korisnicima čije se aktivnosti direktno odražavaju na SUBP. Dvoslojna arhitektura, pored SUBP, mora imati i aplikaciju pomoću koje pristupamo SUBP. Bitno je napomenuti da u okviru ove arhitekture aplikacija i baza funkcionišu izolovano.

Najčešća arhitektura za dizajn aplikacije je **troslojna arhitektura**. Ova arhitektura razdvaja slojeve jedan od drugog po kriterijumu složenosti korisnika i načina na koji oni koriste podatke u bazi. Pored sloja baze podataka i aplikacionog sloja, ova arhitektura uključuje i krajnje korisnike kao entitet (slika 1.3).



Slika 1.4. Troslojna arhitektura informacionog sistema sa aspekta aplikacije

Kao i u ostalim arhitekturama, i u troslojnoj svaki sloj izolovano obavlja svoju ulogu:

- **sloj baze podataka** – na ovom nivou nalazi se baza podataka zajedno sa jezicima za obradu upita; ovde se takođe definišu i odnosi između podataka, kao i ograničenja koja se tiču podataka;
- **aplikacioni sloj** – ovaj sloj obuhvata aplikacijski server i programe koji pristupaju bazi podataka; za korisnika ovaj nivo predstavlja apstraktni prikaz baze podataka;
- **prezentacioni sloj** – krajnji korisnici nisu svesni postojanja baze podataka izvan ovog sloja; na ovom sloju aplikacija može pružiti više različitih prikaza baze podataka (pogleda); svi pogledi koji se na ovom nivou prikazuju generisani su u nižem, aplikacionom sloju.

Arhitektura SUBP može imati i više od tri srodna nezavisna sloja koja se mogu odvojeno modifikovati, ažurirati ili zameniti. Ovakve arhitekture nazivamo višeslojnim.

Rezime

- Način skladištenja podataka zavisi od tipova podataka koji su predefinisani.
- Baza podataka predstavlja zbirku sistemski skladištenih podataka uređenu u cilju jednostavnijih odgovora na upite sistema.
- Koncipiranje, projektovanje i implementacija baze zavise od modela podataka.
- Sistem za upravljanje bazama podataka (SUBP) omogućava fizički prikaz baze podataka u skladu sa odabranom logičkom strukturom, odnosno modelom.
- Arhitektura SUBP (ANSI/SPARC) razlikuje tri nivoa:

    - interni nivo – prikazuje fizički raspored podataka u spoljnoj memoriji;
    - konceptualni nivo – logički koncept baze podataka (šema podataka);
    - eksterni nivo – generisanje lokalnih tipova podataka (pogleda).

- Šema baze podataka logički opisuje podatke celokupne baze, njihova ograničenja i relacije među njima.
- Postoji nekoliko vrsta jezika za komunikaciju sa bazama podataka:

    - DDL – definicija podataka (kreiranje šeme);
    - DML – dodavanje, ažuriranje i brisanje podataka (manipulacija podacima);
    - QL – interaktivno pretraživanje baze pomoću niza izjava (kreiranje upita).

- Najpopularniji integrisani jezik za komunikaciju sa bazama podataka je SQL.
- Najzastupljenija arhitektura aplikacionih programa je troslojna arhitektura, koja podrazumeva:

    - sloj baze podataka – bazu podataka sa definisanim jezicima upita;
    - aplikacioni sloj – server aplikacije sa apstraktnim prikazom baze podataka;
    - prezentacioni sloj – podaci u vidu pogleda po zahtevu korisnika.